Defect Management Through the Personal Software Process

Iraj Hirmanpour
AMS, Inc.

Joe Schofield
Sandia National Laboratories

Software quality improvement begins with defect-free software. The Personal Software Process℠ (PSP℠) defect management framework provides individual software engineers with the tools to prevent and remove defects early in the life cycle. Our experience with the PSP indicates that the application of discipline methods such as PSP provides a mechanism for defect prevention as well as early defect removal and substantial reduction in test time. In this article, we describe the PSP defect management framework and quantitatively demonstrate the reduction of defects by using the PSP defect management methods

Metrics here, metrics there,
metrics metrics everywhere.
ERA and GPA, MPH and MPG;
LDL and HDL, UCL and LCL,
RPM and RBI, BPS and DPI,
upper limits, lower limits,
in-bounds, out-of-bounds,
on schedule, on budget,
out of scope, out of hope!

Can there be any doubt that metricssurround us [1]? Measurement and metrics are foundational for understanding an engineering process. In the software-engineering world, the collection of metrics has been problematic, yet its need persists for process improvement and product quality monitoring. Project measures that predict cost and schedule are easier to obtain and are widely used. However, collecting software defects to measure quality is more difficult and thus not as pervasive as other project measures.

The key measure related to software quality is, of course, defects. According to Watts Humphrey, developer of the Personal Software Process℠ (PSP℠), "The defect content of software products must first be managed before other more important quality issues can be addressed" [2]. Any organizational claims that software quality is improving are unreliable sans defect measures. This article focuses on the PSP defect management system, and reveals how a systematic approach to defect collection and analysis provides individual engineers with the ability to remove defects early in the software development life cycle.

Personal and peer reviews are primary sources of defect detection. Test results are another source of defect detection, albeit a more resource intensive activity. Worse yet, change requests and *trouble reports* are evidence of defects that have made their way to the customer. The PSP's focus on quality software products ameliorates the collateral damage associated with defects discovered by the customer. Despite large investments in testing strategies, the average U.S. software defect removal rate is about 85 percent [3]. These dismal results are the consequence of using less disciplined software-engineering practices that rely on code and test cycles to remove defects.

Given that software engineers inject defects, they should be responsible for identifying and removing them. Our experience with the PSP, supported by the Software Engineering Institute (SEI), indicates that the application of disciplined methods such as PSP reduces the number of defects injected in the process and the amount of test time required to detect and remove them. This reduction is achieved primarily by lowering the number of defects that are introduced and secondarily, by removing defects early in the life cycle rather than in testing. Using the PSP defect management framework, this article will demonstrate how software engineers can improve their defect management process.

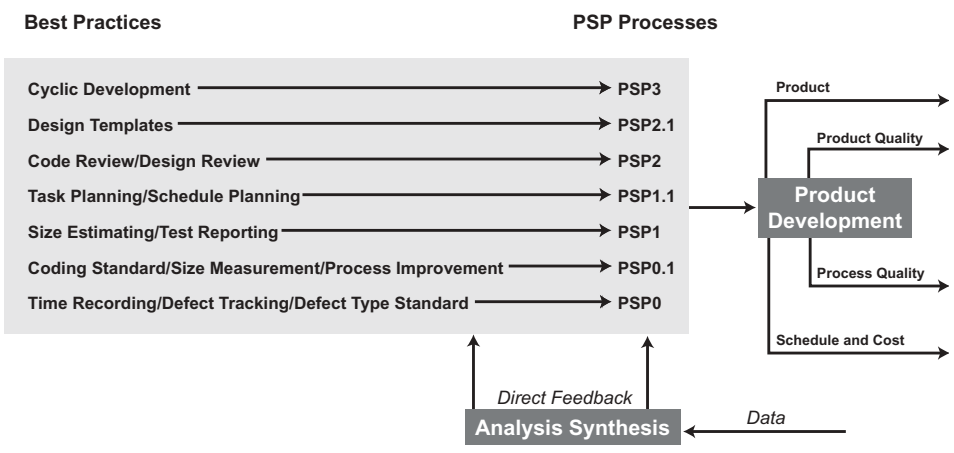
The PSP Framework

The PSP framework is a data-driven feedback system that allows individual software engineers to continuously improve their personal processes by applying statistical process control techniques at the individual level. A PSP practitioner uses a defined software process to apply a set of practices to develop products, while collecting data as part of the development process. Figure 1 illustrates how the collected measurements are used to analyze and assess the impact of a practice on the product and/or process using a feedback loop. This feedback becomes an inherent part of all future product development processes. The framework therefore, offers a road map for collecting data. By analyzing the data, engineers are able to modify their practices and thus improve predictability and quality.

The framework depicted in Figure 1 shows the seven process steps numbered from PSP0 to PSP3. On the left are the new practices that are introduced at that process step. In PSP 1.1 process step, for example, task planning and scheduling planning practices are introduced. It is important to notice that all previous process steps evolved into the schedule planning and tracking process of PSP 1.1. In other words, process steps are evolutionary and cannot be skipped.

A PSP practitioner decides to use one of these process steps to produce software artifacts. The SEI recommends using practices embodied in PSP 2.1 that

Figure 1: *The Personal Software Process Framework*



℠ Personal Software Process and PSP are service marks of Carnegie Mellon University.

Defect Types	
10	Documentation
20	Syntax
30	Build, Package
40	Assignment
50	Interface
60	Checking
70	Data
80	Function
90	System
100	Environment

Table 1: *PSP Defect Types*

inherits all previous practices. The product development process of a PSP practitioner, regardless of which process step is employed, produces two classes of output: the project product and a set of metrics on process and product. Contrast this approach with the classic code and test approach that emphasizes the maturation of the product by removing defects (or discovering requirements) during the test phase.

The PSP Defect Management

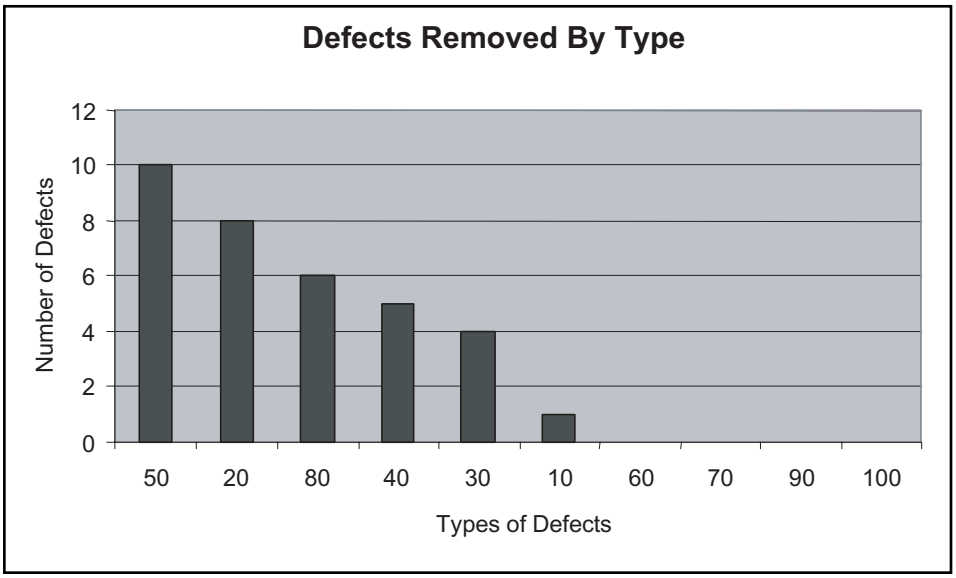
The goal of any defect management process is to eliminate defects from software products. Unfortunately the practice often merely tends to reduce defects [3]. The PSP framework promotes defect management. While learning and practicing the PSP, engineers are required to collect and record data on the process and product during development, including defect data. Starting with the first PSP process step, engineers are introduced to a defect collection method. Within the PSP, a defect is defined as anything that will result in failure of software to operate, causing rework to correct it [4]. The defect collection method consists of establishing a defect classification scheme and recording defect attributes.

For each defect identified, engineers record the defect type based on the classification scheme in Table 1, as well as the following: the injection phase, removal phase, and correction time [5]. To improve a process, it is necessary to know the current state of the process. By writing a program using the first PSP process step, engineers gain insight into their process by exploring the question, "What is my current defect injection/removal rate?"

This defect collection process continues until the fifth step of the PSP during which students would have written seven programs. Once sufficient defect data are collected, engineers are required to determine defect injection and removal rates, and the associated correction time. Armed with this information, engineers produce a design review checklist and a code review checklist based on their personal defect profile. A typical defect profile created by PSP defect data is shown in Figure 2.

Furthermore, the PSP framework provides a structured review process that the engineer follows using the checklist to review his or her work. In the example data shown in Figure 2, the engineer will have defect types 50, 20, and 80 on the checklist because according to personal data, they are injected most often and consume the largest repair time. The fifth process step of the PSP (PSP2) introduces the design review and code review activities as part of the process. The goal is to remove all defects before compiling and testing.

Figure 2: *Defect Profile of a PSP Student*



The PSP review framework consists of process scripts, checklists, and time and defect collection forms. The engineer follows the review script that specifies three phases: review, correct, and check. For each item on the checklist, engineers review each line of design or code from beginning to end. Each time a defect is found it is corrected and its correctness is verified. Time spent fixing the defect and the type of defect are recorded in the defect log. The PSP review process, therefore, is a structured and measured process. The collected review data includes the time spent in review, the number of defects found, the time spent fixing defects, and the number of lines of design or code reviewed.

From these measures, you can derive metrics such as lines of code (LOC) per hour reviewed and defects detected per hour. During the post-mortem phase of the project, two additional metrics are derived called *yield* and *appraisal to failure ratio* (AF/R). Yield is defined as percent of defects removed before the first compile. AF/R is defined as the ratio of percentage of the total time that engineering spent reviewing a product (appraising) and percentage of time that engineering spent compiling and testing a product (correcting failures). An AF/R ratio of two reveals that twice as much time was used to review the product compared to compiling and testing it.

Data gathered during the PSP class is then used to assess the quality of the review and to develop an improvement strategy. Some of the PSP historical data suggests that a review rate must be less than 200 LOC per hour, the yield goal should be around 80 percent, and the AF/R should be greater than the number two.

Unfortunately, the data collected on current practices of software engineers indicates that the opposite is true. Engineers prefer to rush through the coding phase with little or no design, minimize reviews, and then correct defects during the compile/test phase.

The PSP Class Defect Data

As described earlier, PSP students develop 10 programs following progressively evolving practices using the PSP while collecting data on their work. The first seven programs are written using planning, design, code, compile, test, and post-mortem as process phases. Although activities within each phase grow in sophistication, phases stay the

same until program eight, at which time quantitative management practices are introduced and two new phases – design review and code review – are introduced. The expanded and complete PSP consists of the six process phases listed earlier with an additional review phase following both the design and code phases respectively.

The first program is written using the PSP0 process to establish a baseline of current state. Table 2 shows defect data for five PSP classes. Students attending these classes are practicing engineers; all are college graduates. Fifty percent of the students have a master's degree and an average of 11 years experience. As depicted in Table 2, the range of defects varies from 69 to 124 (variation of 55 percent) among classes with an overall average defect rate of 100 per thousand lines of code (KLOC). Similarly, the test defect range varies from 25 to 55 (variation of 45 percent) for each of the five classes and contains an average test defect of 38 defects per KLOC. Data from PSP classes consistently shows a wide variation in performance among software engineers. Variation in defects is no exception.

These data form the baseline from which performance improvement is measured. In addition to helping engineers, this data is also useful to the organization. If this organization were suddenly required to estimate defect injection rate as part of preparing a quality plan, 100 defects/KLOC would be a valid estimate based on historic performance. In lieu of these measures, the organization is void of quantitatively determining its defect profile or the amount of time engineers use to fix the *bugs*.

Once all of the defect management practices are introduced, a sharp drop in both total defects and test defects is achieved. As shown in Table 2 in all classes, the overall average in process defects improved by 50 percent and overall average test defect improved by 63 percent. Since testing removes only a fraction of defects [2], fewer defects discovered in test, while performing similar levels of defect removal, equates to fewer defects in the final product. Measured quality improvements are an additional benefit of following these process steps.

The next question is, "Do engineers who learn and apply PSP in their work processes produce higher quality products than non-PSP trained engineers?" To answer the question, three recent graduates of a PSP class agreed to collect and share data on their projects based on the PSP model. They gathered data on 13 small maintenance projects with a total of 13,914 LOC. As shown in Table 3 on those 13 projects, the total defects per KLOC were 22 and test defects per KLOC were reduced to four.

	Number of Students	Defects Per KLOC Start	Defect Per KLOC End	Test Defect Per KLOC Start	Test Defect Per KLOC End
Class 1	8	69	40	25	9
Class 2	7	108	28	40	11
Class 3	10	83	24	33	21
Class 4	7	124	74	35	10
Class 5	11	119	83	55	17
Average		**100**	**50**	**38**	**14**

Table 2: *Comparison of Defect Profile at Start and End of the PSP Class*

While not a statistically viable study, the similarity between the results over five classes and those from 13 actual projects based on the PSP model reinforces the fact that dramatic improvement can be achieved if graduates continue to follow the PSP process.

Summary and Conclusion

Software quality begins with the removal or substantial reduction of software defects before other quality attributes such as maintainability, portability, reliability, or usability can be considered. A defect is referred to as anything that causes the software not to function as specified and requires efforts to correct it. Needless to say, a major source of software defects is missing or incomplete requirements, which are not addressed in this article and relate to the requirements engineering process.

However, once a set of requirements is agreed upon, the next challenge is to design and build software that satisfies the requirements and is defect free, that is, it functions as specified. Once the requirements are specified, defects enter the product during design and coding phases. Since software engineers are engaged in the design and coding activities during which defects are injected, they should also remove them. The quality principle of *do it right the first time* stipulates that these defects be detected and removed by the engineers while in development and not during test or deployment.

The PSP defect management framework enables software engineers to prevent defects and then to identify and remove injected defects early to avoid costly corrections later in the life cycle. There are two components to defect management: defect prevention and defect detection. The PSP defect data collection system provides the necessary information to use statistical methods to identify the root causes of defects and to develop strategies for preventing defect injection. The PSP's structured and measured review process enables software engineers to detect and remove defects early. The measurements taken during the review are analyzed to improve the efficiency of the review process, thus providing a continuous improvement mechanism.

Our experience with teaching classes and collecting data on students supports the notion that as engineers use PSP defect management practices, their defect injection rate is reduced substantially (defect prevention), and defect removal efficiency (defect detection) is increased resulting in reduced test and repair time. Lower costs and higher customer satisfaction follow naturally.

Metrics abound in the construction of software, as in other engineering disciplines. We have attempted to demonstrate how the use of metrics, in this case defect collection and analysis, contributes to measured improvement in software quality and a reduction in development and support time. Addi-tional benefits accrue to organizations as their software engineers continue to practice the PSP as part of their daily activity. The PSP provides a framework for software process improvement. Its processes can sustain

Table 3: *Average Defect Rates*

	Start of class	End of class	After class
Total Defects/ KLOC	100	50	22
Test Defects/ KLOC	38	14	4

enhanced practices within an organization's software engineering community long after the class has concluded.◆

References

1. International Function Point Users Group, et. al. IT Measurement: Professional Advice from the Experts. Addison-Wesley, 17 Apr. 2002: 221.
2. Humphrey, W. S. A Discipline for Software Engineering. Addison-Wesley, 1995.
3. Jones, Capers. Software Quality. International Thomson Computer Press, 1997: 400.
4. Humphrey, W. S. A Discipline for Software Engineering. Addison-Wesley, 1995: 12.
5. Ibid: 44.

Additional Reading

1. Khajenoori, S., and I. Hirmanpour. An Experiential Report on the Implications of Personal Software Process for Software Quality Improvement. Proc. of the Fifth International Conference on Software Quality, Austin, TX, Oct. 1995 <www.sei.cmu.edu/tsp/recommended-reading.html>.

About the Authors

Iraj Hirmanpour is a principal of AMS, Inc., a software process improvement firm and a Software Engineering Institute Personal Software Process℠/Team Software Process℠ (PSP℠/TSP℠) transition partner. He is a SEI-certified PSP instructor and TSP launch coach. Hirmanpour is also a visiting scientist with the Carnegie Mellon Software Engineering Institute collaborating on the transition of PSP and TSP into academic curricula.

AMS Inc.
421 7th St. NE
Atlanta, GA 30308
Phone: (386) 405-4691
E-mail: ihirman@earthlink.net

Joe Schofield is a technical staff member at Sandia National Laboratories, a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy. He chairs the organization's Software Engineering Process Group, is the Software Quality Assurance Group leader, and is accountable for the introduction of the Personal Software Process℠ and the Team Software Process℠. He has dozens of publications and conference presentations in the software engineering realm and has taught graduate level software engineering classes since 1990.

Sandia National Laboratories
MS 0661
Albuquerque, NM 87185
Phone: (505) 844-7977
Fax: (505) 844-2018
E-mail: jrschof@sandia.gov